Journal of Circuits, Systems, and Computers

World Scientific
www.worldscientific.com

IMPROVED MODIFIED FAT-TREE TOPOLOGY NETWORK-ON-CHIP

Abdelhafid Bouhraoua

Computer Engineering Department, King Fahd University of Petroleum and Minerals,
PO Box 969, 31261 Dhahran, Saudi Arabia
abouh@kfupm.edu.sa

Muhammad E. S. Elrabaa

Computer Engineering Department, King Fahd University of Petroleum and Minerals,
PO Box XXX, 31261 Dhahran, Saudi Arabia
elrabaa@kfupm.edu.sa

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

C-based cycle-accurate simulations are used to evaluate the performance of a Network-On-Chip (NoC) based on an improved version of the modified Fat Tree topology. The modification simplifies routing further and guarantee orderly reception of packets without any loss of performance. Several traffic models have been used in these simulations; Bursty and non-bursty traffic with uniformly-distributed destination addresses and non-uniformly-distributed destination addresses. A simple new traffic model has been developed for generating non-uniformly-distributed destination addresses. This model is general enough to be used in developing new NoC architectures and captures universally accepted place-and-route methodologies. Simulation results are used to illustrate how the hardware resources of a modified Fat Tree NoC can be minimized without affecting the network performance. The performance of a NoC with regular Mesh topology was also evaluated for comparison with the modified Fat Tree topology.

Keywords: Networks-On-Chip, Systems-on-Chip, ASICs, Interconnection Networks, Fat Tree, Routing.

1. Introduction

Recent technology scaling has enabled the integration of larger number of processing elements, computational cores and memories. The complexity of communication between these cores has also increased tremendously. In addition, very short time-to-market constraints has placed higher stress on efficient design methodologies. As a result, the *Networks-on-Chips* (NoCs) paradigm, in *Systems-on-Chips* (SoCs), has emerged as an alternative to ad-hoc wiring or bus-based global interconnecting networks. The idea of turning the on-chip interconnection network into yet another IP block (Intellectual Property block) that is both flexible and scalable is very appealing to the time limitations imposed by the market requirements. The main advantage of this approach is the fact that it constitutes a systematic solution for the common issues of compatibility, bandwidth requirements, and performance. Hence, the general consensus is that the communication

requirements, as well as the design flow of billion-transistor SoCs are best accommodated by shared, segmented interconnection networks [1-3].

There has been a significant amount of effort made in the area of NoCs, and the focus has mostly been on proposing new topologies, and routing strategies. However, recently the trend has shifted towards engineering solutions and providing design tools that are more adapted to reality. For example, power analysis of NoC circuitry has intensively been studied [4, 5], more realistic traffic models have been proposed [6], and more adapted hardware synthesis methodologies have been developed [12].

However, high throughput architectures haven't been addressed enough in the literature [22]. Beside the efforts related to the Nostrum [7] and the Æthreal [8], most of the other efforts , also based on a regular mesh topology, did not achieve throughputs exceeding 30% of the wire speed (defined as 1 *flit/cycle/client*) [7]. A quasi-mesh flexible NoC architecture, x-pipes, was proposed as a way to enhance throughput and reduce power through custom insertion of NoC resources (routers and links) into the final design [14-15]. This custom insertion, however, increased design time (timing closure harder to achieve), demands special design skills from the SoC designers especially to avoid deadlocks and did not increase throughput significantly (~11% improvement in latency over a regular mesh) [15].

Several non-mesh based topologies have also been proposed [16-21]. The Spidergon NoC, a bi-directional ring with cross-connections, was proposed as a fixed and optimized topology that can realize cost-effective interconnection network for MPSoCs [16]. It has low node degree, homogeneous routers (the same router used throughout the whole network), vertex symmetry and simple deterministic routing scheme. Virtual channels are used to eliminate deadlocks. The maximum message hops are N/4, where N is the number of clients on the NoC. It was shown in [17] that even for a small network size of 16-clients, the Spidergon saturates at an effective injection rate of 30% of the wire speed. For a 64-client network, it saturates at 8% injection rate, a worse performance than 2-D mesh NoCs. WK-Recursive networks [18] have also been proposed as on-chip interconnection networks [19-20] for their high degree of regularity, symmetry and expandability to any arbitrary size without the need to reconfigure the edges. Virtual channels are also employed for deadlock avoidance. Simulation results [20-21], however, show that wk-recursive NoCs outperform 2-D mesh for low injection rates only and they saturate earlier than 2-D meshes.

The modified Fat Tree (MFT), proposed in [9, 10] aimed at addressing the throughput issue. The FT topology was modified in order to eliminate contention altogether thus achieving a throughput of nearly a 100% [9]. This result does not come without a price which is mainly the high number of links (wires) at the edge of the network. Performance evaluation of the MFT was carried out in [9, 10] using only non-bursty traffic model with uniform destination address distribution. No performance of other NoCs was evaluated using the same traffic model for comparison. Also, link utilization is a major concern with the MFT since it employees a large number of links. No evaluation of what percentage of these links is actually used during heavy traffic was carried out in [9, 10].

Such results could be used to prune unnecessary links thus reducing hardware cost while preserving high performance. The purpose of this work is summarized in the following points:

1. Modify the original MFT NoC to simplify the routing further and guarantee the orderly reception of packets without any impact on performance,
2. Use more traffic models to evaluate the improved MFT,
3. Compare the performance of the improved MFT other NoCs (mainly the 2-D mesh) using the same traffic models,
4. A major concern with the MFT is that it employees a large number of links. Hence this work also aims to evaluate the maximum percentage of links utilized during heavy traffic. Such results could be used to prune unnecessary links thus reducing hardware cost while preserving high performance.

The network construction, the formal determination of the routing function and the router architecture of the improved MFT are first presented in section II. In section III the traffic models used in the simulations are introduced. This includes a new application-independent model for non-uniform address generation that follows a realistic approach. Simulations results are shown in section IV. These results include latency measurements under different traffic models for several network sizes and types (MFT and Mesh-based networks) and measurements of the maximum link utilization in every router level of the MFT. Finally conclusions are presented in section V.

2. Modified Fat-Tree NoCs

MFT is a new class of NoCs based on a sub-class of Multi-Stage Interconnection Networks topology (MIN). More particularly, a class of bidirectional folded MINs; chosen for its properties of enabling adaptive routing. This class is well known in the literature under the name of Fat Trees (FT) [8]. The FT has been enhanced by removing contention from it as detailed in [10]. Below is a brief description of the FT and MFT network topologies.

2.1. *FT Network Topology*

A FT network, Figure 1, is organized as a matrix of routers with n rows; labeled from 0 to $n-1$; and $2^{(n-1)}$ columns; labeled from 0 to $2^{(n-1)} -1$. Each router of row 0 has 2 clients attached to it (bottom side). The total number of clients of a network of n rows is 2^n clients. The routers of other rows are connected only to other routers. Hence, two clients can be reached from any router in row 0. A router at row 1 is connected downwards to two routers at row 0. This means that it can reach $2x2 = 4$ clients. A router at row 2 is connected to two routers at row 1; thus reaching $2x4 = 8$ clients. So, in general, a router at row r can reach $2^{(r+1)}$ clients.

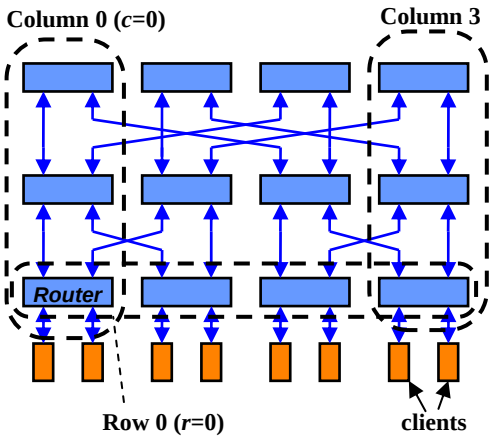
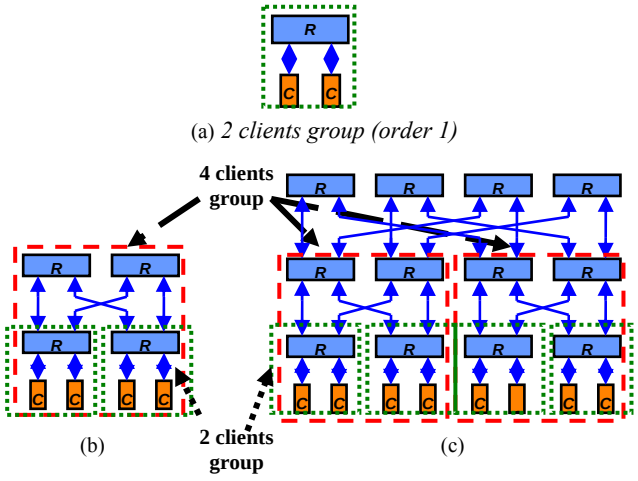


Figure 1: Regular Fat Tree Topology (8 clients)

In a more general view, an FT network is constructed recursively starting from a single router with two clients attached to it, Figure 2. They form the basic building block, called a group (in this case of order 1). Higher order groups are then formed from lower order groups. Figure 2 shows the method for recursively building a group using groups of lower order. It also illustrates how a FT could be scaled up to the next power of 2 (i.e. doubled in size) by replicating the original network as the two halves of the new network and adding a new row of routers of width equal to *2* times the width of the original network on top of them.



(a) *2 clients group (order 1)*

(b) 4 clients group made of 2 two clients groups (order 2)

(c) 8 clients group made of 2 four clients groups (order 3)

Figure 2: FT Recursive Construction.

2.2. *MFT Topology*

Within a single router of an FT network, packet routing falls in one of two cases; routing of packets coming from the bottom links which are either routed up or routed down and downward routing of packets coming from the up links which are always routed down (left or right). Since the number of upward links is equal to the number of bottom links, there cannot be any contention when routing up as explained in section 3.5. Contention, however, might occurs when routing down. Because the bottom links are split in right and left links, deterministic routing of packets will lead to contention. In other words, if several packets coming from the up links need to go right, there will be a contention. One of these packets will earn the right to use the link while the others will be waiting for it to complete as shown in Figure 3.

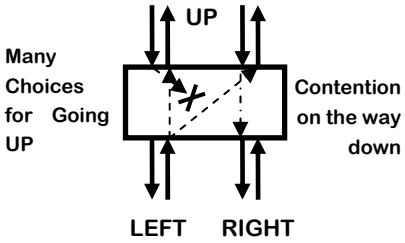


Figure 3 – Contention in FT networks

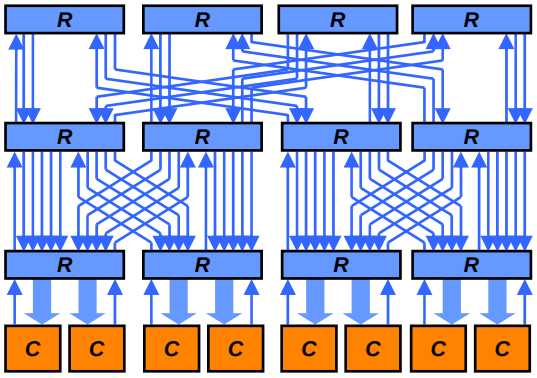


Figure 4 – Modified FT Topology

Contention can be removed if there are enough output ports in a router so that they can accommodate any combination of incoming packets. Since Contention occurs only on the downward path, doubling the number of output ports in the downward direction will eliminate the contention. This is the adopted strategy for the modified FT network [9, 10]. Figure 4 illustrates the doubling of the downward links (the thick downward arrows going out of the routers of row 0 represent multiple links). At each router, the downward output ports (links) are double the number of upper links. Then the input ports of the adjacent router (at the lower level), to which it is connected are also doubled. This is

needed to be able to connect all the output ports of the upper stage router. This will continue till the client is reached where it will have $2^n - 1$ input links. Each of these input links will feature a FIFO buffer as called for by the original MFT architecture [9, 10].

3. Routing in the new MFT

Routing in the original MFTs is similar to that of FTs and simply follows the routing in binary trees. A packet is routed up until it reaches a router that has a path to its destination. This router is called a *summit* for convenience. The FT structure, based on a superposition of binary trees, naturally provides packets with several upward paths. Any upward path will eventually lead to a summit where a downward path to the packet's destination is provided. However, the question of how to route up given that the topology is adaptive in this regard is of practical importance, since it influences other aspects of NoC such as orderly reception of packets per flow. The downward path to the packet's destination is unique as it is the case for any regular tree structure. Hence upward routing in FTs is basically adaptive while downward routing is inherently deterministic. This adaptivity in upward routing has been eliminated from the MFT in this work by routing packets upward along the same side. This ensures the orderly reception of packets without any impact on performance. Another advantage of this is that packets arriving from a specific client will always arrive at the same lane, hence arbitration at the destination client is simplified further (no need to examine source address to determine priority). Detailed description of the routing scheme within the newly developed router architecture is presented below.

3.1. *Client Labeling*

Clients are labeled in an increasing order starting from the left to the right, with all the labels (i.e. the addresses) being within the interval $[0, 2^n - 1]$. The relation between the client address and the column coordinate of row-0 routers is given by the following equation:

$$addr = 2c + s \quad (1)$$

Where the selector $s = \{0, 1\}$, based on the client's position. For the clients connected to the left of row-0 routers, $s = 0$, and for those connected to the right, $s = 1$.

3.2. *Packet Structure*

Packet boundaries are indicated via the use of two flag signals, called the start-of-packet SOP and the end-of-packet EOP signals, respectively. This allows packets of randomly variable-length to be sent over the NoC. The packet data flow in-between the activation of both signals is commenced by a header field, which contains the destination address of the client to which the packet is to be routed, followed by a variable-length data field, which contains the actual communication data, Figure 5. Note that the routing information is all contained within the packet's header.

Figure 5 – Packet structure

For all what follows, Table 1 below groups all the used symbols, their definition, units and range.

Table 1: Symbol Definitions, Units and Valid Range

Symbol	**Definition**	**Unit**	**Valid Range**
n	Total number of rows	*rows*	$[1 .. \infty[$
r	Router *(r, c)* row index	*rows*	$[0 .. n-1]$
c	Router *(r, c)* column index	*columns*	$[0 .. 2^{n-1}-1]$
addr, daddr	Client address client destination address	*clients*	$[0 .. 2^{n} -1]$
G_L	Group of order r reached from router *(r, c)* left link	-	-
G_R	Group of order r reached from router *(r, c)* right link	-	-
C_L	column of the leftmost router(s) in G_L	*columns*	*Constant*
P_L	address of leftmost client in G_L	*clients*	$2c_L$
I_L	router *(r, c)* interval of client addresses on the left downward links	*clients*	$[2c_L .. 2c_L + 2^r -1]$
I_R	router *(r, c)* interval of client addresses on the right downward links	*clients*	$[2c_L + 2^r .. 2c_L + 2^{r+1} -1]$

3.3. *Reach Range*

The first step in building the routing scheme is to determine the range of clients that can be reached by a router *(r, c)* on the downward path. As was described in the network construction section, a router *(r, c)* is connected to two groups of order r; a group G_L to its left with an associated address interval I_L and another to its right, G_R with a corresponding address interval I_R. These groups have the following properties:

- The size of I_L is equal to the size of I_R and is 2^r clients.
- G_L and G_R are adjacent because connected to router *(r, c)* thus I_L and I_R are contiguous intervals and $I_L < I_R$.

The two intervals I_L and I_R are hence computed as follows:

$$I_L = [P_L, P_L+2^r -1] \text{ and } I_R = [P_L+2^r, P_L + 2^{r+1} -1] \quad (2)$$

P_L corresponds to the address of leftmost client in G_L. Since P_L is an address, it can be written as (see Equation 1):

$$P_L = 2c_L + s_L \quad (3)$$

c_L corresponds to the column coordinate of the leftmost router at row *0* of G_L. Since, P_L corresponds to the address of *leftmost* client in G_L, then $s_L = 0$ because client is on the left. Therefore:

$$P_L = 2c_L \quad (4)$$

Hence, the two intervals I_L and I_R become:

$$I_L = [2c_L\ ,\ 2c_L + 2^r - 1] \quad \text{and} \quad I_R = [2c_L + 2^r,\ 2c_L + 2^{r+1} - 1] \quad (5)$$

c_L is computed from the router column coordinate c as follows:

Any router of coordinates *(r, c)* connected to G_L and G_R has its column coordinate c within the interval $[c_L\ ,\ c_L + 2^r - 1]$. This is because the router *(r, c)* belongs to a group of order $r+1$ composed of the two groups G_L and G_R, which makes its lowest column coordinate the same as for G_L. This also means that the number of router columns in this group is $2^{(r+1)-1} = 2^r$. This actually means that:

$$c = c_L + k;\ 0 < k < 2^r. \quad (6)$$

It clearly shows that the value k can be represented using r bits. Thus c_L is obtained by simply clearing the lowest r bits of the column coordinate c.

3.4. *Finding the "Summit"*

A packet is first routed up until it reaches a summit, the first router encountered that provides a path to the packet destination address *(daddr)*. This means that *daddr* is in one of the two intervals I_L and I_R associated with the summit's left and right lower order groups. As long as the packet's destination address *daddr* is outside I_L and I_R, the packet is routed up. To determine if $daddr \in I_L$ or $daddr \in I_R$, is equivalent to satisfying the following inequality:

$$2c_L \leq daddr \leq 2c_L + 2^{r+1} - 1$$

Or

$$0 \leq daddr - 2c_L \leq 2^{r+1} - 1 \quad (7)$$

Therefore, when the double inequality is satisfied, the value $daddr - 2c_L$ is represented with a maximum of $r+1$ bits. Because the lower r bits of c_L (i.e. the lower $r+1$ bits of $2c_L$) are all 0s, representing the value $daddr - 2c_L$ with $(r+1)$ bits requires that the upper $(n-(r+1))$ bits of this value to be all 0s. Consequently:

$$daddr[n-1:r+1] - c_L[n-2:r] = 0$$

$$daddr[n-1:r+1] = c_L[n-2:r] \quad (8)$$

Finally, the condition upon which a summit is found is that the upper $(n - (r+1))$ of *daddr* are equal to the upper $(n-1-r)$ bits of c_L.

3.5. *Routing Upward*

When a packet reaches a router (r, c) from a bottom link, the upper $(n - (r+1))$ bits of *daddr* are compared with the upper $(n-1-r)$ bits of c_L associated with the router, as explained above. If there is a match then this is the summit router and the packet is routed

downward along the other bottom link of the router. If there is no match then the packet is routed upward along the same side (e.g. if it arrived on the left bottom link then it proceeds along the left upper link). This reduces upward routing to a binary decision, a departure from the original MFT routing scheme [1,2]. As such, this ensures packet ordering; packets of an end-to-end flow are received in the same order in which they were sent by the same FIFO at the destination. This simplifies the client's interfaces as well as higher layers of the network protocol. It also simplifies the router's design and does not impact the performance.

3.6. *Routing Downward*

After traversing the summit, the packet is routed downwards (left or right) until it reaches its destination. If $daddr \in I_L$, the packet is routed to the left and if $daddr \in I_R$, the packet is routed to the right.

$daddr \in I_L$, if $2c_L \leq daddr \leq 2c_L + 2^r - 1$

$daddr \in I_R$, if $2c_L + 2^r \leq daddr \leq 2c_L + 2^{r+1} - 1$

This means:

$daddr \in I_L$, if $0 \leq daddr - 2c_L \leq 2^r - 1$

$daddr \in I_R$, if $2^r \leq daddr - 2c_L \leq 2^{r+1} - 1$

The value $daddr - 2c_L$ corresponds to the lower r bits of $daddr$ which are $bits[r-1:0]$. This is because any value of c_L has its $r-1$ lower bits equal to 0s by construction. Thus:
If $0 \leq daddr - 2c_L \leq 2^r - 1$ then $daddr[r-1] = 0$ and if $2^r \leq daddr - 2c_L \leq 2^{r+1} - 1$ then $daddr[r-1] = 1$.

This clearly shows that a single bit of the destination address $daddr$ is sufficient to decide the routing direction.

4. Router Architecture

The router architecture relies on the fact that contention is eliminated along the route to destination, which is achieved, as mentioned above by doubling the links in the downward direction. As a consequence of doubling the links in the downward direction, several models of routers will be present in the network. Models differ from one another by the number of links in the downward path. Routers belonging to the same row will be instances of the same router model. A generic router model has been defined using parameterized Verilog HDL (Hardware Description Language).

The architecture of a router model will comprise:

- two input ports and two output ports for the upward path
- k input ports and $2k$ output ports for the downward path.

two extra downward outputs folding the upward left input to the right and the upward right input to the left (shown in red in Figure 6), to address the case when the router is a "summit".

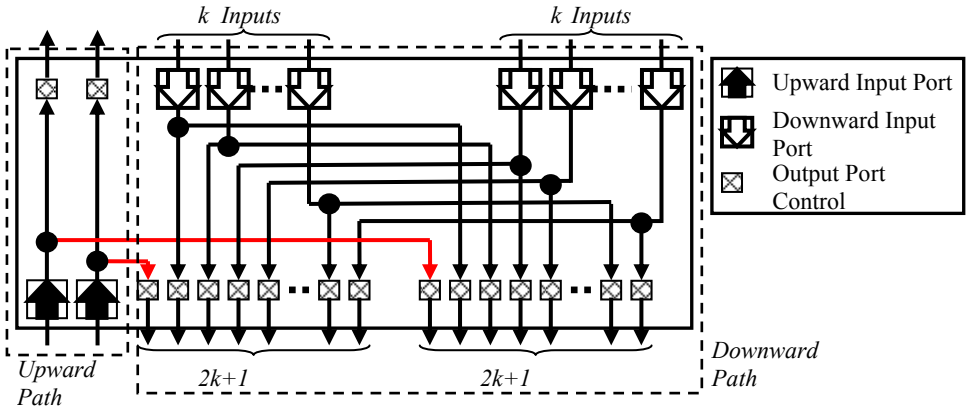


Figure 6 – The adopted router architecture

Figure 6 shows the adopted router architecture with the doubling of the output ports on the downward direction. No internal FIFOs are required since no contention can occur. The internal structures of the different ports making the architecture of the router are shown in Figure 7.The sheer simplicity of the router compensates for the increase in the number of ports. This NoC structure can easily support backpressure implementation as part of the link control logic.

4.1. *Routing Function Circuitry*

Two circuits are used to implement the routing algorithm: one circuit for downward routing, and one for upward routing. The routing function circuits are duplicated for each input port, where either one or both of the two circuits are used, based on whether the input port is directed downward or upward, respectively (this is because downward packets are always routed downward, whereas upward packets have the choice to be routed either way, requiring the two types circuits, cascaded). Note that for each circuit, the packet header is examined and the embodied destination address is passed to the routing circuitry.

Figure 8 show the two circuits forming the upward and downward routing functions. The router's column and row coordinates *c* and *r* are constants.

The upward routing function is centered around comparing the upper bits of the destination address to the upper bits of the router's column *c*. If they are equal, then the router is the summit for the examined packet and the packet is routed downward on the opposite side. However, if they aren't equal then the packet should is routed upward along the same side of the router. Hence the upward routing is basically a binary decision. The downward routing function is also a binary decision (left or right) and is based on the value of the *r*'th bit of the *daddr*. A value of 1 of this bit means that *daddr* is

in the right group of the router and the packet is routed to the right. Similarly a value of 0 in the *r*'th bit of the *daddr* means that *daddr* is in the left group and the packet is routed to the left. This is accomplished through the use of a simple *n-to-1* mux where the router's row *r* selects the appropriate decision bit. It should be noted that the only variable part of the router is the variable length comparator used in the upward routing function and the constants representing the router's coordinates.

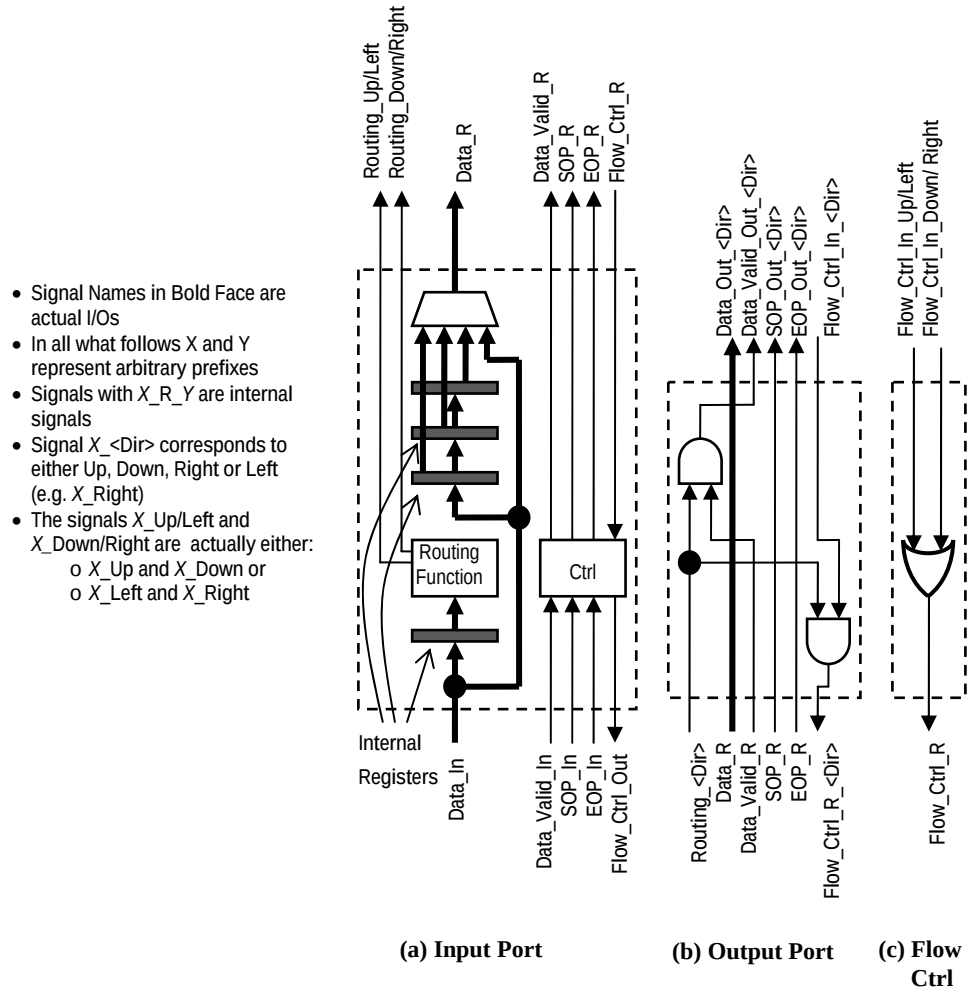


Figure 7: Port Structures

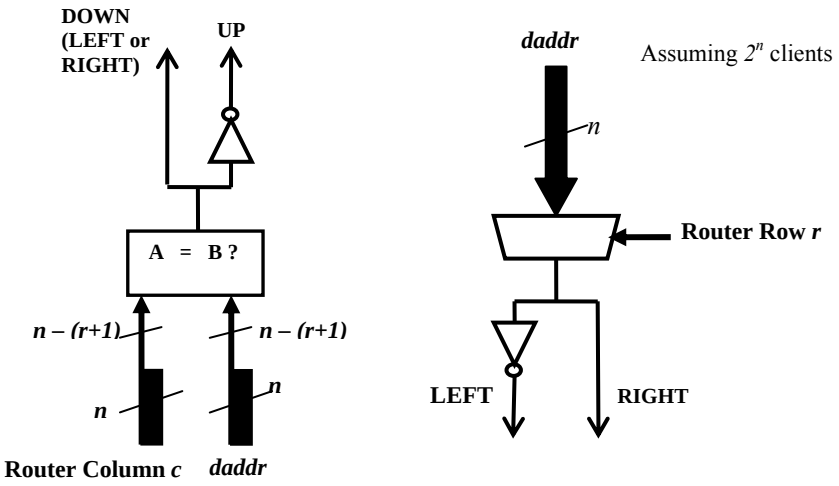
4.2. *Router Parameterization Scheme*

The router parameterization scheme uses the features of the Verilog language to realize parameterization. The model description is mainly based on the use of parameters and the

`generate` construct. As explained above, the routing function itself is mostly the same for all routers except for the variable-length comparator and the constants representing the router's coordinates. These are very easily generated for each router. The router's architecture is very modular with the routing circuitry replicated for each I/O port as explained above.

Figure 9 shows an excerpt of the *Verilog* model. The model has been verified by simulation. The model is built using modules describing the different ports. The code size amount to a total of about 800 lines comprising, the input port, output port, routing function, flow control and router generator.

Also a simple C program has been developed for generating MFT networks using instantiations of the parameterized router model. The router instances parameters are set with the appropriate values for each row of the network. This generator produces a *Verilog* file containing a model of the desired network. The generated file can be directly used for functional simulations and logic synthesis.



(a) Routing Up Circuit (b) Routing Down Circuit

Figure 8: Routing Function

5. Traffic Models

Traffic modeling for NoCs has been a persistent problem. This is in sharp contrast to computer interconnection networks where traffic of different applications has been modeled extensively [23-26]. NoCs' traffic patterns depend on many factors such as the available computing blocks (called IPs or intellectual property blocks), different concurrent applications running at the time, and applications partitioning and placement in different IPs. In [6] a statistical model was proposed to model the traffic of different applications. The proposed model utilizes a three-tuple to model traffic burstiness, hop count, and spatial distribution (likelihood of each processor to send a packet). These

parameters were empirically extracted from trace runs on very specific CMP (Chip Multi-Procesor) platforms with no specific application placement. As shown in [6] these parameters differed greatly from one CMP platform to another. Also, these traces were run in isolation within the simulated platforms as the usual case with CMPs utilization (offloading of computationally expensive applications). Hence the general applicability of this model is very limited.

In this work, traffic modeling is viewed as a way to stress out the NoC and anticipate worst case yet realistic traffic patterns. Traffic generated at each client is independent from the rest and is modeled in terms of two parameters; burstiness (how much the transfer size varies) and distribution of the destination addresses. The later would directly determine the hop count of the packet. At high injection rates the NoC is stressed out greatly.

```
generate for(i=0;i<num_in_ports;i=i+1) begin:prt_dble
        wire [7:0]      prt_in_frm_up_out_data;
        wire            prt_in_frm_up_out_valid;
        wire            prt_in_frm_up_out_sop;
        wire            prt_in_frm_up_out_eop;
        wire            prt_in_frm_up_in_fc;
        wire            prt_in_frm_up_route;

        port_in_frm_up #(r_mask) prt_in_frm_up_inst(
                .clk            (clk),
                .reset          (reset),
                .in_data        (in_up_data[(i+1)*8-1:i*8]),
                .in_valid       (in_up_valid[i]),
                .in_sop         (in_up_sop[i]),
                .in_eop         (in_up_eop[i]),
                .out_fc         (out_up_fc[i]),
                .out_data       (prt_in_frm_up_out_data),

        .sop_out_left           (out_dwn_sop[i]),
        .eop_out_left           (out_dwn_eop[i]),
        .fc_in_left             (in_dwn_fc[i]),
        .route                  (prt_in_frm_up_route));
endgenerate
```

Figure 9 – Excerpt of the parameterized router mode

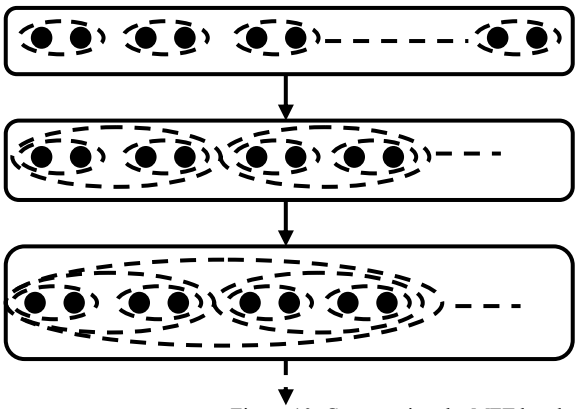
5.1. *Traffic Burstiness*

Two types of traffic were used in the simulations; bursty traffic and non-bursty traffic (where the burst size is fixed at 1 packet). For the bursty traffic, the burst size (in packets) is a random variable with a uniform distribution of a specific range. The range is twice a specified minimum burst size **BZ**. For example if **BZ**= 16 packets, then the range for burst sizes would be from 16 to 32 packets. The packet size **PZ** was kept constant at 64 words and the link size is equal to one word (i.e. a word may be injected every cycle).

The injection rate was varied from 10% to 90% of the wire speed. The injection rate (**RATE**) was set in an indirect manner through the inter-burst gap. This inter-burst gap was randomized using a uniform distribution with an average value **GZ** that is set to yield the required average injection rate after sufficient number of simulation cycles have elapsed using the following formula:

$$GZ = PZ * BZ\ [(1/RATE) - 1] \qquad (9)$$

For non-bursty traffic, **BZ** is simply set to 1. The start of the first burst transmission for each client was randomized and simulations were carried out for at least one million cycles, ensuring that bursts from all clients do overlap one time or another.



Step 1: Clients that communicate the most with one another are placed in the groups of two

Step 2: Then groups of two that communicate the most among themselves are placed in the groups of four

Step 3: Then groups of four that communicate the most among themselves are placed in the groups of eight and so on

Figure 10: Constructing the MFT by clustering the clients

5.2. *Destination Address Generation*

Again two methods were used for generating the destination address for each packet; using a uniform distribution of and a non-uniform distribution. For the first method a uniform discrete probability distribution is used in setting the destination address of each packet with all client addresses having equal probability. This method implies that clients were placed without any consideration of their communication patterns. This is not realistic especially for heterogeneous SoCs where clients communicate more often with some specific clients than others. The uniform distribution, however, is good for stressing out the network since it will cause more packet hops. In the second method destination addresses are generated based on a weighted probability distribution function. It is based on the idea that clients that communicate the most with one another would be placed next to one another. A simple binary clustering method, shown in Figure 10, is assumed to be used in placing the clients. Clients are first clustered in pairs with their most communicating partners. These groups of two (will be referred to as groups of *order 1*) directly correspond to the basic groups used in constructing MFT as explained earlier. Then groups of two clients are clustered with their most communicating group of two to form the groups of four (groups of *order 2*) and so on till the whole MFT is constructed.

The probability model used to generate the destination addresses follows directly from the construction scheme explained above. Two steps are used to generate the address; 1st the group to which the destination client belongs to is calculated using binary weighted probability as follows; the probability that a packet is sent to the other client within the group of order1 is 50%, to a client outside this group but within the next group order is 25%, and to a client in the next group order is 12.5% and so on. The probability mass functions are shown in Figure 11 for several network sizes. Once the destination group is determined the specific client address within the group is generated using a uniform distribution.

As an example, Figure 12 shows the resulting probability mass function for destination addresses for packets originating at client 0 for a 16-client NoC.

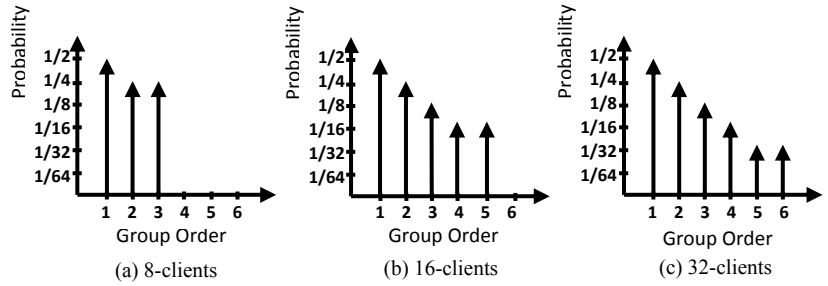


Figure 11: The Probability mass functions used to generate the destination addresses for different NoC sizes. A Logarithmic scale (base 2) has been used for the probabilities for clarity.

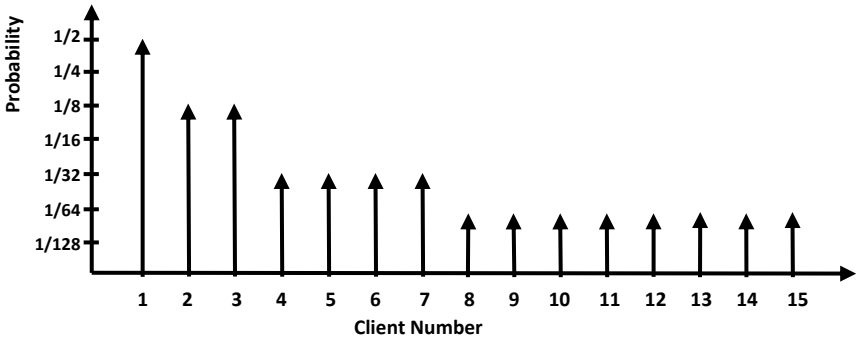


Figure 12: The resulting Probability mass function for the destination addresses for client-0 in a 16-client NoC. Again a Logarithmic scale (base 2) has been used for the probabilities for clarity.

6. Simulation Results

Simulations were performed using a cycle-accurate C-based custom simulator. The packet size (PZ) was fixed at 64 words and the size of the FIFOs at the edge of the network (at the client interface) was fixed at 2K-bits (4 packets). Each client has a total of $2^n - 1$ FIFOs (one per input link as in the original MFT [9]). The emptying rate of these FIFOs is 2 words/cycle. This is the main cause of latency in this type of NoCs since there

is no contention in the routers themselves. Table 2 summarizes the different simulation parameters and simulation conditions.

Table 2: Simulation Parameters

Network Size	16, 32 and 64 clients
Packet Size	64 Bytes
Destination Distribution	Uniform and Local
Offered Load Range	0.1 to 0.9 flits/client/cycle with a 0.1 step
FIFO size	2K-bits
Eject rate	2 flits/cycle
Simulation Duration	10,000,000 cycles

6.1. *Average Latency Measurements*

Latency is defined as the number of cycles elapsing between the time the first word is injected into the network till the last word belonging to the packet is read (i.e. removed) from the corresponding FIFO at the destination. The average latencies versus injection rate are shown in Figure 13 for three network sizes; bursty and non-bursty traffic; and uniform and non-uniform destination address distribution. The offered load is expressed as a percentage of the maximum bandwidth which corresponds to 1 *flit/cycle/client*. The reported latency is the average of all packets' latencies. The following can be observed from these results:

As expected traffic burstiness has increased the average latency noticeably especially at higher injection rates, however none of the simulated NoCs saturated even at the highest injection rate (90%),

In the case of non-bursty traffic, average latency hardly changes with increasing the injection rate for both types of address generation methods,

Uniform address generation actually stressed the network more as evident from the higher average latency for both bursty and non-bursty traffic. This is due to the increased number of hops per packet.

It should be noted that FIFO sizes need not be equal as the case used in the simulations.

In order to compare the MFT's performance to a classical Mesh NoC, another simulator was developed for simulating Mesh NoCs using simple non-bursty traffic with uniform address generation. Figure 14 shows the average latency results for several Mesh sizes. It is very clear that unlike the MFT, these networks start to saturate at very low injection rate (~30%). This is consistent with reported results in the literature for this popular type of NoCs.

6.2. *Link Utilization Measurements*

A major concern with MFT is the over usage of resources, mainly links. The architecture requires the doubling of down links with each level of routers starting from the top level.

The maximum number of links utilized at each router at any instance of time has been collected from all simulation runs (for all types of traffic and injection rates). These results are reported in Figure 15. This figure shows that at lower levels (levels 0 and 1) the maximum number of simultaneously active links is a small fraction of the total links available. Although the obtained results are expected because the MFT has additional links by construction to precisely eliminate the contention source, the potential it offers for savings, especially at the client interface, is huge.

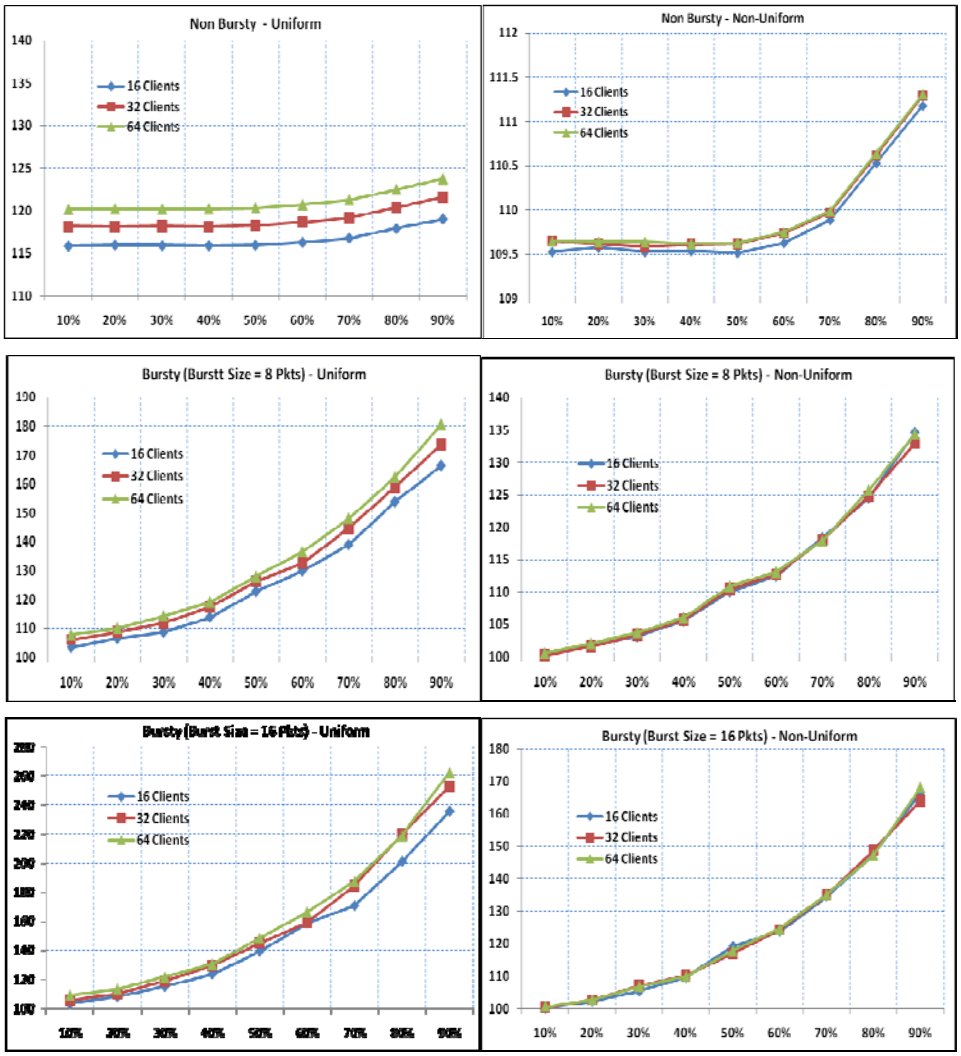


Figure 13: Average latency (cycles) vs. Injection Rate (%) per packet for all types of traffic for three network sizes.

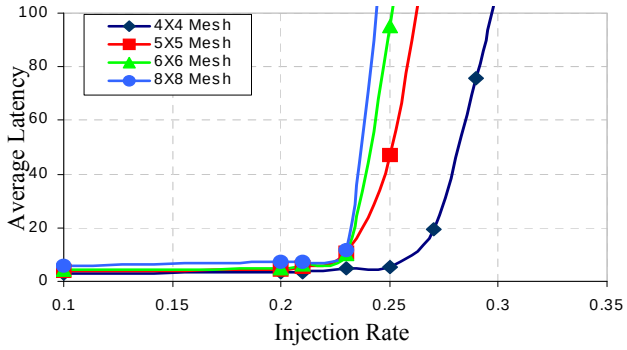


Figure 14: Average Latency versus Injection rate for Mesh-based network of various sizes.

For example for the 64-client network, only 8 out of the 63 available outputs are active at row 0, a merely 12.5% utilization. This constitutes a huge potential for saving resources by eliminating most of the FIFOs at the output interface currently needed to accommodate traffic on the incoming 63 links. However, this would require a modification of the client interface in the form of adding a blocking crossbar. This needs to be investigated in a separate effort.

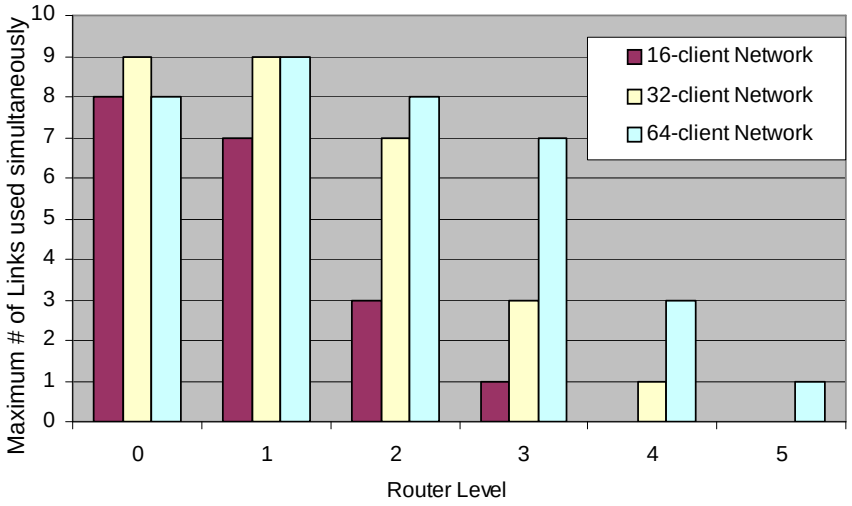


Figure 15: Maximum Link Utilization at any router vs. Router Level for Several networks under all traffic conditions and injection rates.

It can also be noted that maximum number of active links (i.e. FIFOs) at level 0 is around 8~9 for all the networks considered. This can be explained intuitively as follows; as the network size increases there are more clients that could be sending data simultaneously

but there are also more potential receiving clients, hence the number of maximum simultaneous input data streams per client remains the same. This means at most, a client would be almost continuously receiving data streams from 8 different clients simultaneously, far more than any realistic application demands. Hence the potential savings in FIFOs at the clients' interfaces would even be more at higher network sizes.
Destination address distribution significantly affects the link utilization. For non-uniform distribution, where most of the traffic is local within smaller groups, link utilization is much lower than uniform-address distribution. As such, the results reported in Table 3 below are using uniform address distribution. This table shows the percentage of active links per routers level in the MFT. It clearly shows that the downward links from the upper three layers of routers, for every network size, are always active. Besides the simulation results and the obtained throughput, this result can be considered as an additional justification to the doubling of the links strategy adopted in making the MFT.

Table 3: Simultaneously Active links for uniform distribution of destination address.

NoC Size	**Links**	**Router Level**					
		0	**1**	**2**	**3**	**4**	**5**
16	Output links	15	7	3	1	-	-
	Active links	8	7	3	1		
	%	53.33	100.0	100.0	100.0		
32	Output links	31	15	7	3	1	-
	Active links	9	9	7	3	1	
	%	29.03	60.0	100.0	100.0	100.0	
64	Output links	63	31	15	7	3	1
	Active links	8	9	8	7	3	1
	%	12.7	29.03	53.33	100.0	100.0	100.0

However, at the lower router levels, which are closer to the clients, the link utilization percentage starts dropping from 60 to as low as 12.7%. This represents a good indication on the location where the savings on the links should be introduced.

7. NoC Area Estimation

The following equations summarize the MFT components as a function of the number of clients, 2^n:

The total number of routers = $n2^{i-1}$ *(10)*

The total number of input links = $2^n \left(1 + \sum_{2}^{n} 2^{i-1}\right)$ *(11)*

The total number of output links = $2^n \left(1 + \sum_{2}^{n} 2^{i}\right)$ *(12)*

The total number of links = $1.52^n \left(3 + 2 * \sum_{2}^{2n} 2^{i}\right)$ *(13)*

The total number of FIFOs at the input of each client = $2^n - 1$ *(14)*

It should be noted that for a number of clients that is not a power of 2, the MFT structure has to be modified accordingly so that resources (routers and links) are not wasted. The structure is modified as follows: We define a number k such that 2^k will be the nearest power of 2 to n, $n < 2^k$. We also define m as the difference between 2^k and n: $m = n - 2^k$. Then, the modification of the MFT to accommodate n clients is to simply remove all the routers that are used to route traffic exclusively to and from the upper m clients. The ports, within other routers, linking them to these removed routers should also be removed. The resulting network will have less number of routers and links. Figure 16 shows, as an illustration, a network of *11* clients.

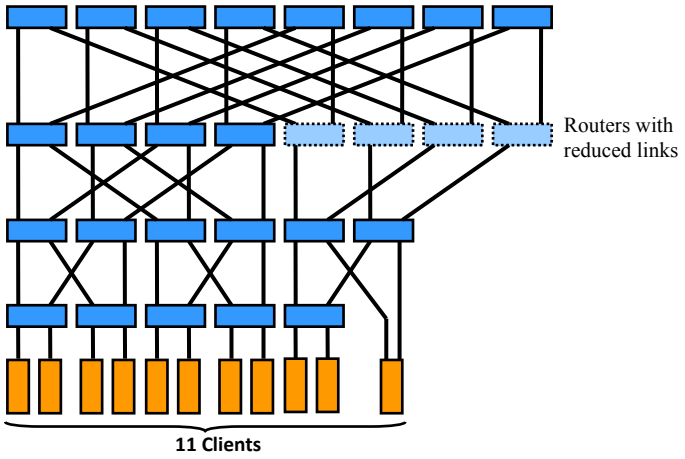


Figure 16: MFT Structure of 11 clients

The fact that the size of the routers is not variable from row to row makes the number of routers of less significance compared to the number of input links which occupy the majority of the area. Therefore, the savings are in number of links and not number of routers because the removed routers are of much larger size than the remaining routers. These routers (dashed outline and light color in Figure 16) have actually lost half their size because of having their right output ports completely trimmed. As a result, routing becomes constant on the downward path as all input ports are directly connected to the left output ports. Routing logic is therefore eliminated resulting is large area savings.

In the example of Figure 16, the 11^{th} client has two links attached to it. This is just to show that the receiving links are coming from two different attached routers. This client can only source packets from the router on the left with its number of receiving links being equal to the number of receiving links of the other clients.

The total gate count of various sized NoCs has been evaluated using a word size of 8-bits by performing logic synthesis of the parameterized RTL-level Verilog description of routers and client interfaces. A breakdown of gate count is shown below:

1. The router is made of:
   - Upward and downward input ports (including routing functions) which have been synthesized using 250 and 260 gates, respectively (mostly FFs).
   - Output ports contain minimum amount of logic (mainly for the backpressure signal)
2. FIFOs in the client interfaces are implemented as embedded SRAMs with a capacity of 4 packets (packet size of 64 bytes) or 2K-bits total size. According to the 2010 International Technology Roadmap for Semiconductors (ITRS 2010) [13], for the latest fabrication technologies of 2009, the area of a 6-transistor SRAM cell with overhead was 0.275 μm^2 and the area of a logic gate in the same technology (also with overhead) was 1 μm^2. As such, the FIFO's SRAM is 550 μm^2 in area, equivalent to 550 gates in the same technology.

Table 4 below shows the gate count for 3 network sizes. The NoC size is dominated by the buffers at the edge of the network. These FIFOs could be trimmed down significantly as explained in section 6 above, thus reducing the NoC area significantly. It should also be noted that even without FIFO trimming, these gate counts represent a small fraction of the chip area (~2.7% for the 64-client NoC on a 1-cm^2 die).

Table 4: Gate count (in K gates) for several NoC Sizes.

NoC Size	Total Gate Count (in KGates)	Estimated area*
16-Clients	**165**	**0.17 mm^2**
32- Clients	**586**	**0.59 mm^2**
64- Clients	**2,689**	**2.7 mm^2**

*Does not include NoC wiring and is based on ITRS's 2010 gate area w/overhead estimate of 1 μm^2

Table 5 below summarizes the major properties of the improved MFT NoC used throughout this work.

8. Conclusions

A simplified and improved version of the modified Fat Tree (MFT) NoC has been extensively simulated using a developed cycle-accurate simulator. All routing has been made deterministic simplifying the router architecture tremendously, ensuring in-order packet reception and resulting in much faster simulations. Simulations were carried out using several traffic models: bursty and non-bursty traffic with different burst sizes, uniform and non-uniform destination address distribution. A new realistic non-uniform

address generation method has been developed. It follows a simple universally acceptable concept of placement of IPs in SoCs. The packet injection rate was varied from 10% all the way to 90% of the wire-speed to stress out the networks being simulated. In contrast to the Mesh topology which saturates at an injection rate of about 30%, simulation results confirmed that the MFT does not saturate under any kind of traffic up to 90% effective injection rate. It also showed that the latency is mainly due to the emptying rate of the FIFOs at the edge of the network at the clients' interfaces. Simulations also showed that if the traffic follows a non-uniform address distribution, then FIFOs can have different sizes. Area estimates based on gate count shows that a NoC based on the improved MFT is very much integrateable.

Table 5: Major Properties of the improved MFT

Property	**Value**
Name	Modified Fat-Tree
Topology	Tree-based
Buffering	At the Network Interface
Routing	Interval-based Tree-based Algorithm
Orderly Delivery	Yes
Description Language	Verilog HDL (model) and C (network)
Lines of Code	Verilog (800) and C (2000)
Flit Size	1 Byte extensible
Packet Header	1 Flit
QoS	Not supported in this revision
Max. Operating Freq.	About 800 MHz. (estimate)
Area (K-Gates) 64 clients	2,700
Routing Latency	Maximum Average of 200 cycles
Throughput	Up to 95% of wire speed

Acknowledgments

This work was supported by King Fahd University of Petroleum and Minerals (KFUPM) through grant # IN070367. Facilities support by King Fahd University of Petroleum and Minerals is highly appreciated.

References

1. L. Benini and G. D. Micheli, “Networks on chips: A new SoC paradigm”, *IEEE Computer*, 35(1):70 – 78, January 2002.
2. W. J. Dally and B. Towles, "Route packets, not wires: On chip interconnection networks", *Proceedings of the 38thDesign Automation Conference*, pages 684–689, June 2001.

3. J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: a scalable, communication-centric embedded system design paradigm," in *Proc. 17th Int'l Conf. VLSI Design*, 2004, pp. 845-851.
4. E. Nilsson and J. Öberg, "Reducing power and latency in 2-D mesh NoCs using globally pseudochronous locally synchronous clocking", *Proc. CODES+ISSS* 2004, pp. 176 - 181.
5. E. Nilsson and J. Öberg, "Trading off power versus latency using GPLS clocking in 2D-mesh NoCs", ISSCS 2005, Vol. 1, pp. 51 - 54.
6. V. Soteriou, H. Wang and L. Peh, "A Statistical Traffic Model for On-Chip Interconnection Networks", in Proceedings of the 14th IEEE Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '06), Sept. 2006, pp 104-116.
7. E. Nilsson. "Design and Implementation of a hot-potato Switch in a Network on Chip". Master's thesis, Royal Institute of Technology, IMIT/LECS 2002-11, Sweden, June 2002.
8. K. Goossens, J. Dielissen, A. Radulescu, "Æthereal network on chip: concepts, architectures, and implementations", IEEE Design and Test of Computers, Volume 22, Issue 5, Sept.-Oct. 2005 pp 414 – 421.
9. A. Bouhraoua and M. E. S. Elrabaa, "A High-Throughput Network-on-Chip Architecture for Systems-on-Chip Interconnect," *Proc. of the International Symposium on System-on-Chip* (SOC06), 14-16 November 2006, Tampere, Finland, pp 1-4.
10. A. Bouhraoua and M. E. S. Elrabaa, "An Efficient Network-on-Chip Architecture Based on the Fat Tree (FT) Topology", *Special Issue on Microelectronics*, *Arabian Journal of Science and Engineering , Dec. 2007I,* pp 13-26.
11. C. Leiserson, "Fat-Trees: Universal Networks forHardware-Efficient Supercomputing", *IEEE Transactions on Computers*, *Vol. C-34, no. 10*, pp. 892-901, October 1985.
12. S. Murali et. al., "Designing Application-Specific Networks on Chips with Floorplan Information", *Proc. IEEE/ACM ICCAD'06, 2006,* pp 355-362.
13. International Technology Roadmap for Semiconductors (ITRC) 2010. http://www.itrs.net/Links/2010ITRS/Home2010.htm
14. F. Angiolini, P. Meloni, S. Carta, L. Raffo,and L. Benini, "A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs," IEEE Trans. Comput.-Aided Design Integrated Circuits Syst., Vol. 26, no. 3, pp. 421–434, 2007.
15. D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Beninic, and G. De Micheli, "Network-on-Chip design and synthesis outlook," Integration, the VLSI journal, Vol. 41, no. 3, pp. 340–359, May 2008.
16. M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: a novel on-chip communication network," *Proceedings of International Symposium on System-on-Chip*, 2004.
17. M. Moadeli, A. Shahrabi, W. Vanderbauwhede, and M. Ould-Khaoua, "An Analytical Performance Model for the Spidergon NoC," *Proceedings of 21st International Conference on Advanced Information Networking and Applications* (AINA'07), 2007, P. 1014.
18. G. D. Vecchia and C. Sanges, "A recursively scalable network VLSI implementation," *Future Generation Computer Systems,* 1988*,* 4(3), pp. 235-243.
19. S. Suboh, M. Bakhouya, J. Gaber, and T. El-Ghazawi, "An interconnection architecture for network-on-chip systems," *Telecom. Systems*, vol. 37, no. 1-3, pp. 137–144, 2008.
20. S. Suboh, M. Bakhouya, and T. El-Ghazawi, "Simulation and evaluation of on-chip interconnect architectures: 2d mesh, spidergon, and wk-recursive networks," *Proc. NoCS*, 2008, pp. 2005–2006.
21. D. Rahmati, A. E. Kiasari, S. Hessabi, and H. Sarbazi-Azad, "A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips," *Proceedings of International Conference on Computer Design* (ICCD 2006), 2006, pp. 142 – 147.
22. Freitas H.C., Navaux P. "A High Throughput Multi-Cluster NoC Architecture", *11th IEEE International Conference on Computational Science and Engineering*, July 16-18, 2008 Sao Paulo, Brazil, pages 56-63

23. I. Y. Bucher and D. A. Calahan. "Models of access delays in multiprocessor memories" *IEEE Transactions on Parallel and Distributed Systems(TPDS)*, Vol. 3, No. 3, pp. 270-280, May 1992.
24. Y.-H. Lee et al. "Consecutive requests traffic model in multistage interconnection networks" *In Proc. of the International Conference on Parallel Processing (ICPP'91)*, pp. 534-541, Aug. 1991.
25. S. Turner, "Performance analysis of multiprocessor interconnection networks using a burst-traffic model", *Ph.D. Dissertation Thesis, University of Illinois at Urbana-Champaign*, 1995.
26. J.V. Luciani,; C.Y.R. Chen, "An Analytical Model for Generalized Traffic Patterns in Finite Buffered Multistage Interconnection Networks", *In Proc. of GLOBECOM'94,* vol. 2, pp. 1065-1069.